Digitization of a Game: "Papel y Boli"

Pau Leyva García

Resum—Aquest projecte té com a objectiu la digitalització del joc de taula anomenat "Papel y Boli", un joc de taula que es juga per equips en què els participants han d'endevinar paraules escrites pels propis jugadors, seguint regles específiques que varien a cada ronda. Per assolir aquesta digitalització, s'ha desenvolupat una aplicació web que permet reproduir totes les funcionalitats del joc físic en un únic dispositiu, eliminant la necessitat de materials com papers, bolígrafs o temporitzadors, elements que limiten els moments en que es pot jugar. Amb la digitalització del joc, l'usuari pot configurar els equips, introduir paraules, controlar el temps amb un cronòmetre integrat i portar el recompte automàtic de punts. Aquesta solució incrementa notablement l'agilitat i disponibilitat del joc, fent-lo accessible des de qualsevol dispositiu amb navegador. El sistema s'ha desplegat en una màquina virtual per permetre l'accés remot a través d'Internet, i ha estat validat amb èxit en diversos entorns i dispositius.

Paraules clau—Digitalització de jocs tradicionals, aplicació web, joc col·laboratiu, experiència d'usuari, Vue.js, Node.js, desplegament, NGINX, arquitectura client-servidor.

Abstract—This project aims to digitize the team-based board game "Papel y Boli", in which players must guess words written by the participants themselves, following specific rules that change with each round. To achieve this, a web application has been developed that replicates all the functionalities of the physical game within a single device, eliminating the need for materials such as paper, pens or timers—elements that typically limit when the game can be played. With the digital version, users can configure teams, enter custom words, control time with an integrated timer, and automatically track scores. This solution significantly increases the game's agility and availability, making it accessible from any browser-enabled device. The system was deployed on a virtual machine to allow remote access via the Internet and was successfully validated across various environments and devices.

Index Terms—Traditional game digitization, web application, collaborative gameplay, user experience, Vue.js, Node.js, deployment, NGINX, client-server architecture.

1 Introduction

TRADITIONAL board games have served for decades as a source of entertainment and social interaction, especially in family environments and among groups of friends. However, their reliance on physical materials and the need for prior preparation can hinder spontaneous use, particularly in contexts where the necessary resources are not readily available.

One game that exemplifies this situation is "Papel y Boli", a cooperative team game in which participants must guess words written by the players themselves. The game unfolds over multiple rounds with specific rules that change in each phase. Despite its popularity, the setup process can be slow and impractical, as it involves writing multiple words on small pieces of paper, shuffling them, preparing a timer, manually counting points, and managing team turns.

In order to preserve the essence of the original game while improving accessibility and setup efficiency, this project proposes its digitalization through an interactive web application. The solution enables gameplay from a single browser-enabled device, eliminating the need for physical materials and offering a faster and more accessible user experience.

The remainder of this paper is structured as follows: Section 2 presents the defined objectives; Section 3 outlines the project's context and state of the art; Section 4 describes the methodology followed during development; Section 5 presents the obtained results; and Section 6 concludes the work and highlights possible future directions.

2 Objectives

The main objective of this project has been to develop an interactive web application to digitize the traditional game "Papel y Boli", preserving its original mechanics while improving accessibility and ease of use. This challenge involved transferring a fully analog activity into a digital environment, without losing the essential elements that make it fun and dynamic for participants. Additionally, the aim was to ensure a fluid, agile, and practical experience that would allow a game to be started within minutes from any location.

- *Contact email: pauleyvagarcia@gmail.com*
- *Mention: Information Technologies*
- *Supervised by: Miguel Hernández Cabronero*
- *Academic year: 24/25*

To make this possible, the following specific objectives were defined and achieved:

- Reproduce the basic mechanics of the physical game: team creation, word input, turn and round system, timer, and scoring system.
- Ensure the entire game could be played from a single web browser-enabled device, eliminating the need for physical materials.
- Design an intuitive interface, understandable for both regular players of the game and new users.
- Ensure compatibility across different devices and browsers.
- Develop a backend that enables communication between the frontend and a database.
- Allow custom game configuration: number of teams, player names, and number of words per person.
- Implement a system to automatically shuffle the submitted words and distribute them randomly.
- Integrate an editable timer to manage the time of each round.
- Establish an automatic scoring system based on correctly guessed or skipped words.
- Perform tests to verify the system's functionality and compatibility.
- Collect user feedback.
- Prepare the production environment and deploy the application on a virtual machine configured for external access, ensuring proper functionality outside the local environment.

In addition to the technical and functional goals, this project has also been oriented toward the consolidation and demonstration of the skills acquired during the Computer Engineering degree. In this sense, the project aimed to achieve the following educational goals:

- Apply the knowledge acquired to design, implement, and validate a complete application, ensuring code quality, maintainability, and efficiency.
- Tackle an open-ended and undefined problem autonomously, learning to analyze requirements, define the system scope, and make reasoned technical decisions by integrating diverse technologies and approaches.
- Develop the ability to integrate frontend and backend within a modern web architecture, correctly managing the communication between components and data persistence via a database.
- Demonstrate the capacity to manage the full development cycle of a system: from UI and UX design to implementation, testing, and deployment.
- Apply best practices in usability, accessibility, and user-centered design to ensure an optimal experience aligned with user needs.
- Show autonomy and responsibility in time management, solving technical problems, and self-learning new tools and technologies when needed.
- Deepen knowledge related to networking, systems, and remote environment configuration, including static IPs, network services, server setup, and reverse proxy configuration.
- Conceptualize and represent the system's architecture and deployment in a structured way using technical diagrams, to facilitate global understanding of how it works, its components, and their interactions.

3 State of the art

The digitization of traditional games is a growing trend in the field of entertainment applications. Games like Monopoly [1], Pictionary [2], or Heads Up [3] have been successfully adapted to digital environments, offering mobile or web versions that preserve their original mechanics. In the realm of word games and team dynamics, examples like Taboo [4] already have digital versions that automate parts of the experience.

However, no existing digital application has been found that fully covers the "Papel y Boli" game. This reveals a clear opportunity to digitize a well-known activity that has not yet been technologically explored. The presented project was born from the will to fill this gap, offering a playable, customizable, and accessible version from any device with a web browser.

To make this possible, well-known technologies were chosen for their efficiency, ease of use, and compatibility. For the frontend, Vue.js [5] was selected—a modern framework that allows for code organization into reusable components and facilitates a fluid, responsive user experience. Although alternatives like React [6] and Angular [7] were considered, Vue offers a gentler learning curve and a simpler structure for individual projects.

For the backend, Node.js [8] was used as the runtime environment and Express [9] as the framework to define HTTP routes easily. The backend acts as an intermediary between the user interface and the database, handling requests related to games, words, scoring, and more. This approach enables separation of concerns and simplifies future scaling or maintenance.

The database chosen was SQLite [10], due to its lightweight nature and ease of integration. While technologies like PostgreSQL [11] offer greater scalability, SQLite is particularly suitable for applications with a low data volume and few concurrent users. This project does not expect more than 50 to 100 simultaneous users, nor a high frequency of writes within short timeframes, ensuring optimal performance. Moreover, SQLite is widely used in many local or lightweight web applications with similar requirements and offers a solid alternative without the need to deploy a separate database management system.

In the area of code validation and quality assurance, automated testing was implemented for both frontend and backend. For the frontend, Jest and Vue Test Utils [12] were used to validate components in isolation via API mocks and simulated user actions. For the backend, Jest was combined with Supertest to verify API routes and their behavior under different scenarios, including errors and edge cases. Integration tests were also conducted, putting frontend components in contact with a live backend to ensure coherence and proper communication between both parts

of the system.

Lastly, instead of using external deployment services like Vercel [13], Netlify [14], or Railway [15], a self-managed deployment was chosen using a virtual machine with Ubuntu. This environment was configured with PM2 [16] to keep server processes running and NGINX [17] to manage routes and serve production files. This setup enabled the application to be opened through an external URL via router port forwarding, thus ensuring accessibility from any network without depending on third-party platforms.

This combination of technologies and approaches positions the project as an original and functional solution that fills a clear gap in the landscape of digitized games. The system has been built using realistic, scalable tools well-suited to the expected use case, maintaining a good balance between simplicity, control, and technical quality.

4 METHODOLOGY

The development of this project was structured following a waterfall methodology, a classic and sequential strategy that enabled each phase of the work to be approached in an organized and controlled manner. Each stage of the project was clearly planned before moving on to the next one, ensuring a solid foundation prior to progressing in functionality.

Initially, alternative methodologies were considered, such as Scrum or incremental development—more commonly used in collaborative environments or projects with evolving requirements. Although their advantages in terms of flexibility and continuous feedback were acknowledged, the waterfall model was chosen for its natural fit with an individual project, with well-defined requirements and a clearly sequential workflow. This choice facilitated clear organization, phase delimitation, and better time management in an academic context with specific deadlines and deliverables.

Nonetheless, a degree of flexibility was maintained, allowing the initial plan to be adjusted to meet real needs and the dynamics of the development process—for instance, temporarily postponing the application of visual styles to prioritize functionality, or splitting the testing phase into two stages: manual and automated.

The planning was divided into eight major blocks (T1–T8), each with clear objectives and weekly-defined tasks. This plan served not only as a guide but was also fundamental for efficiently managing time and ensuring the completion of all project functionalities within the expected timeframe. Additionally, the prior definition of these blocks made it easier to assess the project's status at any given moment.

The structure of this section follows the phases of the waterfall model, and within each phase, the corresponding project stages (T1–T8) are described to accurately reflect the work carried out. Furthermore, two additional sections are included to describe the documentation phase and the adjustments and adaptations made during the process.

Main Stage	Start Date	Fecha de Finalización
T1 – UI/UX Design	March 3	March 23
T2 – Development Environment Setup	March 24	March 30
T3 – Frontend Development	April 1	April 6
T4 – Backend Development	April 7	April 20
T5 – Integration and Game Logic	April 21	May 18
T6 – Testing and Debugging	May 19	June 1
T7 – Deployment	June 2	June 8
T8 – Documentation	June 9	June 22

Fig. 1. Summarized Gantt chart showing the main stages and their chronology.

4.1 Requirement Analysis

Before starting the development of the application, a requirement analysis process was carried out to clearly identify what was needed to effectively digitize the traditional game. This analysis made it possible to understand the limitations of the physical format (such as reliance on paper, manual preparation, or game interruptions) and to define the goals that the digital version had to achieve: improving speed, ease of use, and accessibility, while maintaining the original game's essence.

This initial process included the conceptualization of the digital gameplay flow, the identification of basic use cases (creating a game, entering words, managing turns, controlling time, displaying results, etc.), and the definition of functional and non-functional requirements such as temporary game storage, interface usability, or compatibility with mobile devices.

Moreover, this analysis was complemented by an initial visual approximation of the project, through sketches and basic interface designs, which helped envision how these requirements would be translated into specific screens and interactions. Therefore, the project stage "**T1: UI/UX Design**" is considered part of this analysis phase, as the early design was key to validating the feasibility of the requirements and guiding the rest of the development.

4.2 System Design

The design phase was fundamental to establish the conceptual and visual foundations of the project before beginning its implementation. This stage focused on defining the application's structure, both functionally and visually, and allowed for the anticipation of key development needs, ensuring a coherent and fluid user experience.

First, the general system architecture and deployment diagram were defined, outlining the communication between different parts of the project: the frontend developed with Vue.js [6], the backend using Node.js [9] and Express [10], and the SQLite [11] database. This architecture was conceived for a local application used by a single group on one device, but with global access through web deployment. This initial planning allowed the system to be properly structured, establishing routes, connection points between components, and future deployment requirements.

Next, the visual and functional interface design was carried out, divided into several subphases:

T1 – UI/UX Design (weeks 1-3):

- **T1.1: Initial interface design:** early layout ideas for visual elements—such as buttons, colors, and overall structure—were sketched by hand to gain a global view of the interface.
- **T1.2: Creation of wireframes and prototypes:** based on the sketches, digital wireframes were created using Paint and later with Figma [18], following mobile screen dimensions. This enabled the representation of main use cases and navigation between screens through interactive prototypes.
- **T1.3: Usability validation and adjustments:** Small navigation tests were conducted with users to validate the experience, and improvements were made (e.g., additional buttons or visual reorganization) based on feedback.

This iterative approach to design allowed a clear and solid foundation to be established before functional development began, ensuring that all screens aligned with the game's objectives and offered an intuitive navigation experience for users.

4.3 Implementation

The implementation phase involved the concrete development of the system based on the previously established design. During this stage, the project's functional modules were built and the technical structure required to bring the application to life was set up.

T2 – Development environment setup (Week 4): before beginning coding, the local development environment was configured. Development was conducted locally using Visual Studio Code [19] to maximize efficiency during the early phases, postponing the virtual machine setup until the final deployment stage. During this phase, the version control repository was initialized on GitHub [20], necessary tools were configured to begin coding, and the initial folder structure was established to separate backend and frontend components.

T3 – Frontend development (Week 5): In this phase, the main structure of the frontend was prepared. Based on the design planning and Figma [18] prototypes, the actual development of the views using Vue.js [6] began. Screens were created for the main use cases: Home, Team Configuration (CU01), Word Entry (CU02), Pre-Game, Game (CU03 and CU06), Turn Change (CU04), and Results (CU05). This provided a basis for the eventual implementation of the game logic and backend integration. Vue Router [21] was also configured to enable navigation between screens.

T4 – Backend development (Weeks 6–7): similarly to the previous phase, the general backend structure was prepared and the server was configured using Node.js [9] and Express [10]. During this stage, a comparison was made between possible database solutions—PostgreSQL and SQLite [11]—with SQLite ultimately chosen for its simplicity and suitability for environments with few concurrent users. As a result, the database was set up and connected to the backend.

This phase established the technical foundations that enabled the subsequent integration between frontend and backend, ensuring a scalable and coherent structure for the following development phases.

4.4 Integration

The integration phase was one of the most crucial in the project, as it connected all parts of the system to produce a functional and cohesive product. This stage focused on implementing the game logic and ensuring proper communication between the frontend, backend, and data base making sure the system worked correctly as a whole.

T5 – Integration and game logic (Weeks 8–12): during this period, the functional core of the game "Papel y Boli" was developed, integrating all planned functionalities and ensuring their coordination. Specifically, the following tasks were completed:

- **Frontend-backend communication:** initial communication tests were conducted using HTTP requests to validate correct data transmission between client and server.
- **Implementation of game logic:** the full game flow was programmed, including game creation, player word entry, team-based turn management, thematic round handling, time control with an editable timer, word distribution, and automatic score calculation.
- **Visual refinement and style application:** once the basic functionality was implemented, the visual style defined in the original design was applied, adjusting margins, typography, colors, and layout to achieve a cohesive and attractive result.

This stage required extensive coordination between system components and marked the project's maturity point. The resulting stability allowed confident progress toward the testing and deployment phases.

4.5 Testing

The testing phase played a key role in validating the correct functioning of the application at visual, functional, and structural levels. Following a progressive strategy, this stage was divided into two distinct sub-phases: an initial phase focused on manual gameplay validation and a second focused on implementing automated tests to ensure robustness and facilitate future improvements.

T6 – Testing and debugging (Weeks 12–13):

- **T6.1 – Manual testing and functional validation:** during the first week of the testing phase, complete functional validation was carried out through real gameplay sessions simulating different scenarios with various combinations of players and teams. This allowed:
  - Verifying smooth navigation between screens.
  - Confirming correct creation and configuration of games.
  - Validating the timer, turn changes, and round rotation behavior.
  - Checking accurate recording of guessed and skipped words, as well as score calculation.

  Additional tests were conducted using Thunder Client [22] to ensure that all API (backend) routes were functioning correctly and returning expected responses.
- **T6.2 – Automated testing:** in the second week,

automated tests were implemented and organized into three levels:

- **Frontend unit tests:** Using Jest [16] and Vue Test Utils [17], individual frontend components were tested in isolation, simulating API responses to validate visual behavior and local logic.
- **Backend unit and API tests:** using Jest in combination with Supertest [15], server routes and their interaction with the SQLite [11] database were validated using mocks.
- **Integration tests (frontend ↔ backend):** Vue components connected to the real API were tested to ensure correct data loading and handling, validating integration between system layers.

This phase enabled the detection and correction of minor errors, improved code coverage, and established a stable foundation for the final deployment. Additionally, separating manual and automated testing facilitated progressive validation aligned with the project's schedule and priorities.

4.6 Deployment and Maintenance

The deployment of the application was a key phase to ensure its availability in a real environment, accessible from external networks. It involved the complete preparation of a dedicated infrastructure.

T7 – Deployment (Week 14): for this deployment, a personal virtual machine running Ubuntu Server [23] was created and manually configured to gain full control over the process. This decision allowed for the acquisition and demonstration of practical knowledge in system administration and web application deployment.

- **T7.1 Virtual machine and environment setup:** the first deployment stage involved creating and configuring a virtual machine with Ubuntu Server using VirtualBox [24]. Appropriate resources were assigned to ensure stable performance (e.g., 4 GB RAM and 2 CPUs), and the minimal Ubuntu Server version was installed, avoiding unnecessary graphical environments to optimize resource usage. During installation, basic options were configured such as user creation, network setup, and the installation of the OpenSSH [25] server to enable remote access.
  After installation, a static IP address was configured by editing the Netplan configuration files to ensure that the machine would always be reachable within the network. Essential deployment tools were then installed: Git [20] for code management, Node.js [9] and npm to run the backend, PM2 to keep the backend running, and NGINX [26] to serve the frontend and act as a reverse proxy.
- **T7.2 System deployment and server configuration:** once the environment was ready, the project repository was cloned, and both frontend and backend dependencies were installed. The frontend, developed with Vue.js [6] and Vite [27], was compiled in production mode, generating the corresponding static files. These files were placed in the /var/www/ directory, and a new NGINX server block was configured to serve them. Additionally, all API requests to /api were redirected to the local backend using the reverse proxy configuration. This setup was validated through local tests, and several syntax errors in configuration files were resolved before activating the final setup.
  As for the backend, the Express [10] server was started with PM2, ensuring it ran correctly and was set to start automatically upon future virtual machine reboots. Basic network security measures were also applied using UFW (Uncomplicated Firewall), allowing only the necessary traffic for the application's operation.
- **T7.3 External access and final validation:** once the system was running locally, the home router was configured to redirect port 80 to the virtual machine's IP, making the application accessible from the Internet. This setup allowed full system testing in an external environment, simulating real usage. Access tests were conducted from mobile networks and external computers to validate that the frontend loaded, backend communication worked, and the overall application behaved correctly.

This phase not only completed the project with a functional and publicly accessible instance but also provided valuable hands-on experience in server configuration, networking, security, and production environment management.

4.7 Documentation

As the final phase of the project, documentation played a crucial role in consolidating, organizing, and presenting all the work carried out during development. This stage not only allowed the recording of technical decisions and results obtained but also the preparation of necessary materials for formal evaluation and effective communication with the academic committee.

T8 – Documentation (Weeks 15–16): Time was finally dedicated to writing this report and preparing the final documentation for the Final Degree Project. This involved structuring the content clearly and compiling evidence of functionality and testing. Additionally, all development phases were reflected, and the value of the technical decisions was emphasized.

4.7 Adjustments and Adaptations During the Process

Thanks to the detailed and methodical structure established from the project's outset, it was possible to continuously monitor the development's progress, identify critical points, and apply strategic adjustments as needed. This ability to analyze and adapt contributed significantly to the project's overall success.

The following adjustments were made during development, compared to the original plan:

- Deployment of the virtual machine was postponed to focus efforts on completing the core functionality and ensuring its stability before publication.
- Deployment of the virtual machine was postponed to focus efforts on completing the core functionality

and ensuring its stability before publication.
- The testing phase was reorganized, using part of the first week to finalize the visual design and distributing the rest of the time between manual and automated testing, thereby improving efficiency and system validation coverage.

This structured yet flexible approach was key to adapting the project to real needs as they arose and ensuring solid progress without major deviations or compromises in quality. The planning served as a guide and control tool, ensuring that all objectives were met within the established deadlines and with appropriate technical maturity.

5 Results

The results obtained throughout the development of this project reflect the full achievement of the defined objectives and highlight the effectiveness of the chosen methodology.

The final product is a functional, stable, and complete web application that successfully digitizes the traditional "Papel y Boli" game in a faithful and efficient manner.

The results obtained are both technical and functional and fully meet the initial project goals. Users can create, personalize, and play a complete game session from any browser, using only one device, removing the need for physical materials.

These results can be grouped into two major categories: the deliverables produced and the validation actions carried out to ensure their quality.

5.1 Deliverables and Functional Implementation

The project has resulted in a set of products and outcomes that represent the tangible result of the work carried out. In addition to the web application itself, which allows a complete digital game experience, several technical and support materials were produced, all of which were essential for development, validation, and final delivery preparation:

- **Fully functional web application** deployed in production, accessible from any browser and device through a public URL.
- **Complete GitHub repository** with version control for the entire source code.
- **Wireframes and interactive prototypes** created using Figma, simulating full navigation and serving as the foundation for frontend design.
- **System architecture and deployment diagram**, illustrating the communication flow from the user's browser (client) through the Internet and router port redirection to the virtual machine, including the role of NGINX as reverse proxy and frontend (Vue.js) server, its interaction with the backend (Node.js/Express) for API requests, and access to the SQLite database.
- **Server configuration scripts**, specifically for NGINX (frontend delivery, reverse proxy) and PM2 (backend process management), ensuring stable execution and efficient system management in production.
- **Project documentation**, generated in parallel with development, including this final report as a consolidated result.

Regarding the functional implementation, the main milestone achieved was the complete implementation of the game logic, allowing real game sessions to be simulated without any external intervention. The digital version maintains the original structure of the physical game while improving the overall experience through digital enhancements. Therefore, it fully implements all functionalities defined in the use cases and required to play "Papel y Boli" comfortably and flexibly:

- **Custom game creation:** users can choose the number of teams and players, assign names to teams and members, and set how many words each participant will input.
- **Guided word input:** each player inputs their words in the predetermined order, using a screen specifically adapted for the task.
- **Full round and turn management:** the system automatically handles player turns, turn changes, round transitions, and game completion.
- **Dynamic word display:** words are shown randomly during each round, without repetition within the same round.
- **Automatic scoring system:** correct guesses increase the team score, while passes decrease it.
- **Configurable timer:** players can pause, reset, or modify the duration of the timer during the game, offering added flexibility.
- **Final results screen:** at the end of the game, a screen displays the winning team and the scores obtained by each.
- **Extra screens:** although not included in the original use cases, extra screens were implemented to enhance the user experience and improve playability. These include a pre-game context screen for the first player and a home screen to start game configuration.

All these functionalities allow a full game session to be played: with all rounds, personalized teams, user-entered words, visible scores, and no critical errors or need for manual intervention.

Another notable result is the architectural design, based on a clear and effective client–server architecture that separates concerns and ensures maintainability and scalability of the system.

The system is composed of two main modules:

- **Frontend**, developed in Vue.js, follows the MVVM (Model–View–ViewModel) pattern. The .vue templates serve as the View, the <script setup> block acts as the ViewModel, and the data models obtained represent the Model.
- **Backend**, built using Node.js and Express, follows the MVC (Model–View–Controller) pattern. The SQLite database and its management (in database.js) represent the Model, the route definitions in server.js

act as the Controller, and the JSON responses serve as the View for the RESTful API.

5.2 System Validation, Stability, and User Experience

After completing the full implementation of the application, an exhaustive validation process was carried out to verify its stability, accessibility, and user experience in real-world contexts. The goal was to ensure that the platform not only worked technically as expected but also offered a robust, reliable, and well-received experience for end users.

Stability and Real-Time Behavior Results

Once functional implementation was completed, the system's behavior was tested under real usage conditions through multiple full game sessions with different parameter combinations (teams, players, words, and rounds). These tests revealed:

- All functionalities operated smoothly, with correct screen transitions, consistent game state updates, and stable logic flow.
- The system responded properly to edge cases, such as empty fields, attempts to skip when no words remained, or rapid navigation between screens.
- The configurable timer was tested under multiple scenarios (pause, reset, duration change), always maintaining expected behavior without any desynchronization.

These results demonstrated that the system is stable and ready for sustained and diverse usage.

Testing and Automated Validation Results

To reinforce the validation process, both manual and automated tests were implemented across backend and frontend layers:

- **Manual testing:** full game sessions simulated different usage flows and scenarios. These helped detect and fix errors that could affect gameplay.
- **Thunder Client API testing:** GET, POST, and PUT requests were executed to ensure data consistency and correct interactions with the database.
- **Backend automated tests (Jest + Supertest):** all API routes were tested using mocked database scenarios to simulate real usage conditions.
- **Frontend automated tests (Jest + Vue Test Utils):** individual Vue components were tested in isolation with mocked API responses to confirm proper UI logic.
- **Integration tests** between frontend and backend ensured that live API calls behaved correctly and data was handled as expected across layers.

This multi-layer testing approach ensured extensive system validation and quality assurance.

Deployment and Global Accessibility Results

After confirming all functionalities in the development environment, the application was successfully deployed in a production environment to ensure global accessibility and optimal performance.

The platform is hosted on a dedicated Virtual Machine (VM) running Ubuntu Server 24.04 LTS, configured with a static IP address (192.168.1.105) within the local network.

To manage deployment and ensure stable runtime, the system uses NGINX as a reverse proxy and static content server, and PM2 for backend (Node.js) process management.

The application is fully accessible from any location via Internet. This was achieved by configuring port forwarding in the router, mapping the public WAN port 80 to the internal VM's port 80. As a result, the system is reachable through the dynamically assigned public IP (e.g., http://83.43.181.95).

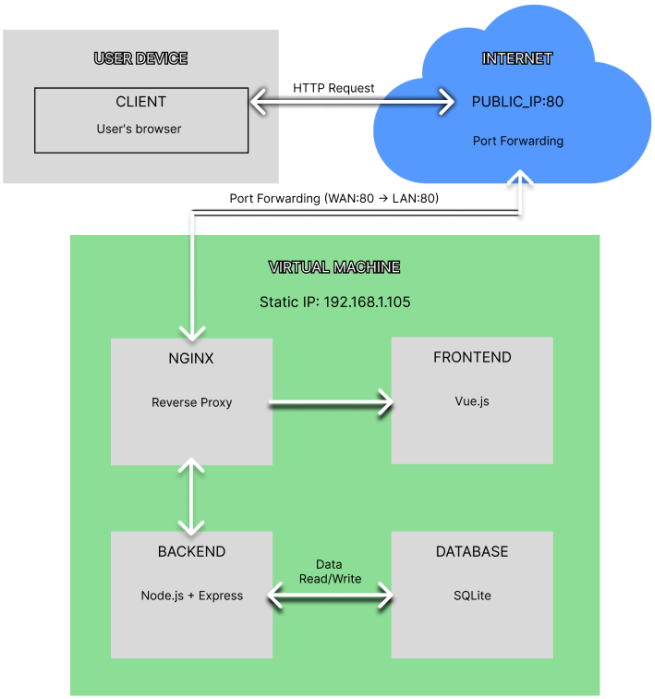

Fig. 2. Web Application Architecture and Deployment Flow in Production Environment. Shows the interconnection between the client, the Internet, the Virtual Machine, and its components (NGINX, Vue.js Frontend, Node.js/Express Backend, and SQLite Database), detailing the communication flow and the role of the reverse proxy.

This architecture, as described in the system deployment diagram, reflects full control over network and infrastructure configuration.

The accessibility and correct operation were verified from:

- **Mobile devices:** Tested on smartphones and tablets using 4G/5G networks, under both Android and iOS.
- **External computers:** Access tested from laptops and desktops connected to external Wi-Fi networks.
- **Various browsers:** Compatibility confirmed on modern web browsers such as Google Chrome, Mozilla Firefox, and Microsoft Edge.

In all tests, the application performance and load times were comparable to local use, and all functionalities worked without incident, confirming the robustness of the deployment.

Real User Feedback

To complete system validation, qualitative feedback was gathered through test sessions with real users. Participants included friends, colleagues, and family members—some familiar with the physical game and others without prior exposure.

The evaluation methodology involved asking participants to play a game session and then share their impressions in an informal but detailed way through oral questions about usability, playability, and comparisons with the traditional version.

This oral format encouraged open feedback, resulting in valuable insights on usability, perceived benefits, and areas for improvement. Key findings included:

- High fidelity to the physical game, maintaining its essence while improving flow and convenience.
- Clear and intuitive interface, with a clean and functional design that supported spontaneous use.
- Improved agility in starting a game, eliminating manual setup tasks like paper handling and timers.
- Positive reception of the editable timer, which was highlighted as a genuinely useful feature during gameplay.

This evaluation confirmed that the application meets not only its functional requirements but also real user expectations, offering a rich, efficient, and well-received digital version of the game.

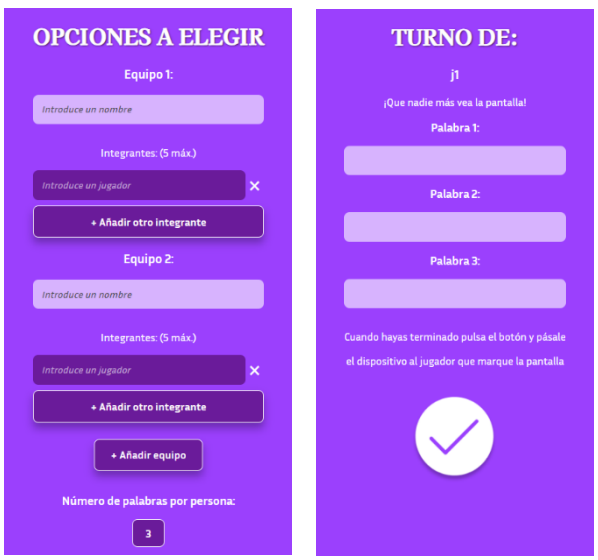

Fig. 3. Team Configuration and Word Entry screens. These views correspond to the CU01 and CU02 use cases, allowing players to set up their teams and enter words before starting the game.

6 Conclusions

The development of this Final Degree Project has resulted in the full achievement of the initial objectives. A functional, intuitive, and robust web application has been successfully created to digitally reproduce the traditional game "Papel y Boli", making it fully usable in real-world contexts with an optimized user experience. Furthermore, the system was correctly deployed in a production environment accessible via the Internet, demonstrating both its technical feasibility and the author's mastery of the entire development and deployment process.

One of the most noteworthy aspects of the project is its coherence from conception to final deployment. The project emerged from a real and relatable need—modernizing and simplifying the way "Papel y Boli" is played—which allowed for the definition of clear goals and sustained motivation. The methodological approach and task organization were key to the successful development, and having independently managed all phases of the project significantly contributed to both technical and personal growth.

Throughout the project, multiple technical challenges were addressed: configuring the server and network for real deployment, using version control systems, developing the frontend with Vue.js, backend with Node.js and Express, UI design with Figma, implementing automated tests using Jest and Supertest, and validating the system manually across multiple environments. Best practices were also followed in documentation, prototyping, planning, and task tracking.

Regarding the system's structure, the choice to operate from a single shared device was deliberate. Rather than a limitation, this design decision preserved the face-to-face essence of the original game. It effectively solved common issues of the physical format, such as manual setup, point tracking, and time management. This digital version automates those tasks, allowing users to start and play matches quickly and smoothly. Moreover, this solid foundation opens the door to future enhancements, such as a multi-client version for online gameplay with independent devices.

From a learning perspective, the project enabled the application and consolidation of existing knowledge, while also acquiring new skills in areas such as service deployment, network security, interface design, time management, and process optimization. Additionally, it fostered the development of essential professional competencies: planning ability, autonomy, organization, adaptability, and continuous self-evaluation.

Finally, the developed system is fully functional and practically useful for any group of users wishing to play "Papel y Boli" easily and efficiently. Its use is particularly well-suited for social, educational, or recreational environments, where a collaborative activity that blends tradition with technology is desired. This project thus provides a real-world digital solution, easily scalable, and with potential for future evolution into more advanced versions.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Miguel Hernández Cabronero, for his continuous guidance, constructive feedback, and support throughout the development of this Final Degree Project. His mentorship has been key to effectively shaping the project and achieving a successful outcome.

I would also like to thank all the individuals who tested the application and provided valuable feedback. Their collaboration was essential in improving the user experience, identifying potential enhancements, and validating the system's performance in real-world scenarios.

REFERENCES

[1] "Monopoly," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Monopoly_(game)
[2] "Pictionary," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Pictionary
[3] "Heads Up!" App Store. [Online]. Available: https://apps.apple.com/us/app/heads-up/id623592465
[4] "Taboo (game)," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Taboo_(game)
[5] Vue.js, "The Progressive JavaScript Framework." [Online]. Available: https://vuejs.org
[6] React, "The library for web and native user interfaces." [Online]. Available: https://react.dev
[7] Angular, "One framework. Mobile & desktop." [Online]. Available: https://angular.io
[8] Node.js, "Node.js Documentation." [Online]. Available: https://nodejs.org
[9] Express, "Fast, unopinionated, minimalist web framework." [Online]. Available: https://expressjs.com
[10] SQLite, "SQLite Home Page." [Online]. Available: https://sqlite.org
[11] PostgreSQL, "The world's most advanced open source Relational Database." [Online]. Available: https://www.postgresql.org
[12] Jest, "Delightful JavaScript Testing." [Online]. Available: https://jestjs.io
[13] Vue Test Utils. [Online]. Available: https://test-utils.vuejs.org
[14] Vercel, "Develop. Preview. Ship." [Online]. Available: https://vercel.com
[15] Netlify, "The platform for modern web projects." [Online]. Available: https://www.netlify.com
[16] Railway, "Deploy infrastructure, faster." [Online]. Available: https://railway.app
[17] PM2, "Advanced, production process manager for Node.js." [Online]. Available: https://pm2.keymetrics.io
[18] NGINX, "NGINX Open Source." [Online]. Available: https://nginx.org
[19] Figma, "Collaborative interface design tool." [Online]. Available: https://www.figma.com
[20] Visual Studio Code, "Code editing. Redefined." [Online]. Available: https://code.visualstudio.com
[21] GitHub, "Where the world builds software." [Online]. Available: https://github.com
[22] Vue Router, "The official router for Vue.js." [Online]. Available: https://router.vuejs.org
[23] Thunder Client, "REST API Client for VS Code." [Online]. Available: https://www.thunderclient.com
[24] Ubuntu Server, "Secure, fast and powerful server OS." [Online]. Available: https://ubuntu.com/server
[25] OpenSSH, "OpenBSD Secure Shell." [Online]. Available: https://www.openssh.com
[26] NGINX, "NGINX Official Website." [Online]. Available: https://www.nginx.com
[27] Vite, "Next Generation Frontend Tooling." [Online]. Available: https://vitejs.dev